HAAGA-HELIA
ammattikorkeakoulu

Kompakti asiakkaiden hallinta, automaattinen virheiden raportointi ja turvallinen maksusuoritus

Juho Mehtonen

Opinnäytetyö
Tietojenkäsittelyn koulutusohjelma
2014

HAAGA-HELIA
ammattikorkeakoulu

Tiivis-

telmä

Päiväys
25.10.2014

Tekijä(t)
Juho Mehtonen

Koulutusohjelma
Tietojenkäsittely

Opinnäytetyön otsikko
Kompakti asiakkaiden hallinta, automaattinen virheiden raportointi ja turvallinen maksusuoritus

Sivu- ja liitesivumäärä
33 + 3

Opinnäytetyön otsikko englanniksi
Compact customer management, automatic error reporting and secure payment

Opinnäytetyö käsittelee Glitchware Oy:n pilottisovelluksen jatkokehittämistä projektina, lisäten toimintoja ja ominaisuuksia olemassa olevaan sovellukseen, jotta voidaan tarjota kompakti asiakkaiden hallinta, automaattinen virheiden raportointi ja turvallinen maksusuoritus.

Opinnäytetyö koostuu teoria- ja projekti-osiosta. Teoria-osa käsittelee teknistä taustaa ja hyödynnettäviä menetelmiä. Projekti-osio käsittelee työn rajausta, jonka avulla määriteltiin vaadittuja ominaisuuksia ja niiden sisältöä, ominaisuuksien toteuttamista ja niissä hyödynnettyjä menetelmiä ja ratkaisun testaamista yleisellä tasolla.

JQuery:n, AJAX:n ja tarkoin suunniteltujen SQL-lauseiden ansiosta ratkaisu on optimoitu sekä kompakti ja siinä on asiakkaiden hallinta, joka mahdollistaa asiakkaiden lisäämisen ja muokkaamisen käyttäjäystävällisesti, automaattisen virheiden raportoinnin, joka toimii pohjana jatkossakin jokaiselle luodulle metodille ja turvallinen maksuprosessi, jossa hyödynnetään ajan tasalla olevia standardeja.

Projekti edistää merkittävästi pilottisovelluksen vientiä markkinoille, koska pilottisovellus kykenee nyt turvallisiin maksusuorituksiin ja tukee siten yrityksen liiketoimintaa.

Asiasanat
Testaus, Integraatio, Ohjelmointi, Java, JavaScript

HAAGA-HELIA
University of Applied Sciences

Abst-

ract

Päiväys
Date

Author(s) Juho Mehtonen	
Degree programme Business Information Technology	
Report/thesis title Compact customer management, automatic error reporting and secure payment	**Number of pages and appendix pages** 33 + 3

This study examines the further development of the pilot application by Glitchware Oy as a software development project, adding functionality and features to an existing application in order to provide compact customer management, automatic error reporting and secure payment.

The study consists of a theory overview and a description of the project development process. The theory overview discusses technical background and methods applied in the study. The project development section deals with defining features, content, implementation, utilized methods and general testing.

JQuery, AJAX and thoroughly planned SQL statements make the solution optimized and compact. The final solution has a customer management feature which supports user-friendly account management, automated error reporting as a basis for new methods, and a secure payment process utilizing the latest standards.

Implementing this project has significantly lowered the barrier for launching the pilot application to commercial markets thanks to its secure payment transaction features.

Keywords
Testing, Integration, Programming, Java, JavaScript

Sisällys

1 Johdanto

Opinnäytetyö käsittelee yrityksen pilottisovelluksen jatkokehittämistä projektina, lisäten toimintoja ja ominaisuuksia olemassa olevaan sovellukseen. Tavoitteena on yksinkertaisesti saada helppo ja toimiva ratkaisu asiakkaiden hallinnasta, automaattisesta virheiden raportoinnista ja turvallisesta maksusuorituksesta. Projekti on tavoitteena toteuttaa nopealla aikataululla sen tärkeyden vuoksi. Tiukka aikataulu ei saa vaikuttaa toteutettavien ominaisuuksien laatuun.

Projekti on rajoitettu projektin toimeksiantajan laatimien toiveiden mukaan. Asiakkaita tulee pystyä lisäämään, muokkaamaan ja listaamaan. Asiakkaan tili ja käyttäjät tulee pystyä tarvittaessa jäädyttämään. Asiakkaan käyttäjiä tulee pystyä listaamaan, oikeuksia vaihtamaan, määrää lisäämään ja asiakkaan tilin pääkäyttäjän salasana tulee pystyä resetoimaan. Virheen tapahtuessa virheraportin tulee automaattisesti lähteä sähköpostilla projektin toimeksiantajan sähköpostiin. Lisäksi tulee pystyä suorittamaan maksu turvallisesti ja menetelmän tulee olla helposti päivitettävissä uudempaan versioon.

Opinnäytetyö tehdään Glitchware Oy:lle toimeksiantona. Projekti on viimeinen suuri askel yritykselle ennen sovelluksen vientiä markkinoille ja sen tehtävänä on yksinkertaistaa ja nopeuttaa liiketoiminnan kannalta hyödyllisiä prosesseja, jotta resurssit voidaan keskittää markkinointiin. Jatkokehitettävä sovellus on ennestään tuttu, joten on mielenkiintoista olla mukana kehittämässä sitä jälleen eteenpäin. Kompaktin asiakkaiden hallinnan puuttuessa asiat tehdään hitaasti käsin, joka ei ole liiketoiminnallisesti kannattavaa. Ilman automaattista virheiden raportointia virheiden selvittäminen vie tahattomasti aikaa ja maksuprosessin puuttuessa liiketoiminta kärsii.

Tekninen tausta tullaan käymään tarkasti läpi, jotta pystytään antamaan mahdollisimman tarkka kuva tehdystä työstä. Ilman teknistä taustaa ei myöskään olisi mahdollista ymmärtää kuinka ominaisuudet on toteutettu. Projektissa hyödynnetään Javaa ohjelmointikielenä, koska pilottisovellus on toteutettu sillä. Tutuksi tullut JavaScript ja siihen pohjautuvat tekniikat JQuery ja AJAX ovat todella hyödyllisiä ketterän sovelluskehityksen kannalta ja siksi ehdottomia tässä projektissa. Prosessi lähtee liikkeelle haastattelun muodossa tehtävästä vaatimusmäärittelystä, jossa kootaan käyttötapauksiin tarkka selonteko ominaisuuden toiminnallisuuksista ja käyttötiheyksistä. Käyttötapauksilla pyritään selvittämään ominaisuuden tärkeyden ja tarvittavien testien määrän. Tarvittavien käyttötapauksien jälkeen käydään läpi tarkasti tarvittavat menetelmät sekä arvioidaan niiden tärkeyttä ketterän sovelluskehityksen kannalta, jotta projekti saadaan päätökseen vaaditussa aikataulussa. Lo-

puksi ominaisuudet toteutetaan tärkeysjärjestyksessä ja ne testataan kattavasti toiminnallisuuden takaamiseksi. Opinnäytetyö on todella laaja ja keskittyy ominaisuuksien suunnitteluun ja toteuttamiseen, jolloin opinnäytetyön painotus ei ole testauksessa.

1.1 Keskeiset käsitteet

Input-kenttä	Määrittää tekstikentän, johon käyttäjä voi kirjoittaa tekstiä. (W3Schools 2014a)
Data	Määrittää esimerkiksi input-kentästä haettua tietoa.
Metodi	Tapa, jolla määritellään miten objektia käsitellään. (Oracle 2014a)
Objekti	Määrittää muuttujan, jolla on tila, esimerkiksi nimi ja käyttäytyminen, kuten tyhjä. (Oracle 2014b)
Bugi	Sovellus tekee jotain mitä sen ei pitäisi tehdä. (Patton 2005, 13-15)
Serialisoituminen	Hyödyntää jQueryn serialize-menetelmää.
Tietokanta	Säännönmukainen kokoelma dataa. (Kofler, 4)
Annotaatio	Mahdollistaa sellaisen datan tuonnin sovellukseen, joka ei ole varsinaisesti osa sovellusta. (Oracle 2014c)

2 Tekninen tausta

Projektiin on valittu ketterän sovelluskehityksen kannalta tarpeellisia tekniikoita, kuten JavaScript ja siihen perustuvia tekniikoita ja kirjastoja. JavaScriptiä on helppo hyödyntää, koska JavaScript-tulkki löytyy jokaisesta nykyaikaisesta web-selaimesta, jolloin sen hyödyntäminen on todella helppoa (Peltomäki 2000, 6).

Säilytettävä data on aina altis hyökkäykselle, jonka lopputuloksena varastettu data joko paljastuu tai sitä muokataan. Varastetulla datalla voidaan varastaa identiteettejä ja tehdä paljon harmia. Jos data ei ole salattu sitä voi lukea suoraan, jolloin sen saaminen on melko vaivatonta, jonka vuoksi projektissa hyödynnetään hash-menetelmää. (Gates, 300).

2.1 JavaScript

Skriptikieliin kuuluva JavaScript on suunniteltu muun muassa sovellusten ja toimintojen laajentamiseen. JavaScript ei vaadi erikseen kääntämistä, joten sen hyödyntäminen on helppoa ja se edistää ketterää sovelluskehitystä (Peltomäki 2000, 6).

JQuery on JavaScriptin kirjasto, jonka avulla JavaScriptin kirjoittaminen on helppoa. JQuery:n avulla on mahdollista tehdä dynaamisia verkkosivuja eikä käyttäjä välttämättä huomaa edes, että sivua päivitetään sen silmien edessä ja voidaan sanoa, että se mahdollistaa reaaliaikaisen kommunikoinnin käyttäjän kanssa antamalla validoinnin yhteydessä käyttäjälle ilmoituksen virheellisesti syötteestä. JQuery mahdollistaa myös AJAX:n yksinkertaisen käytön osana sovellusta. JQuery:llä voidaan myös muuntaa tarvittava tieto lähetettävään muotoon hyödyntäen serialize-menetelmää. JQuery-tekniikan käyttäminen on joissain tapauksissa hieman yliammuntaa, mutta projekti on osana suurempaa kokonaisuutta, jolloin se on välttämätön (Otero & Larsen, 3-5).

2.2 JSON

JSON on JavaScript-kieleen pohjautuva tekniikka, jota hyödynnetään data-objekteiden välittämiseen. JSON:lla välitetään dataa serverin ja sovelluksen välillä sellaisessa muodossa, että ihminen voi sen lukea (JSON 2014a).

Käsiteltävä data ei ole aina oikeassa muodossa, jolloin JSON:lla joudutaan kääntämään annettu data oikeaan muotoon hyödyntäen JSON-stringify-menetelmää, jolloin vain yksi osa datasta käännetään oikeaan muotoon (JSON 2014b).

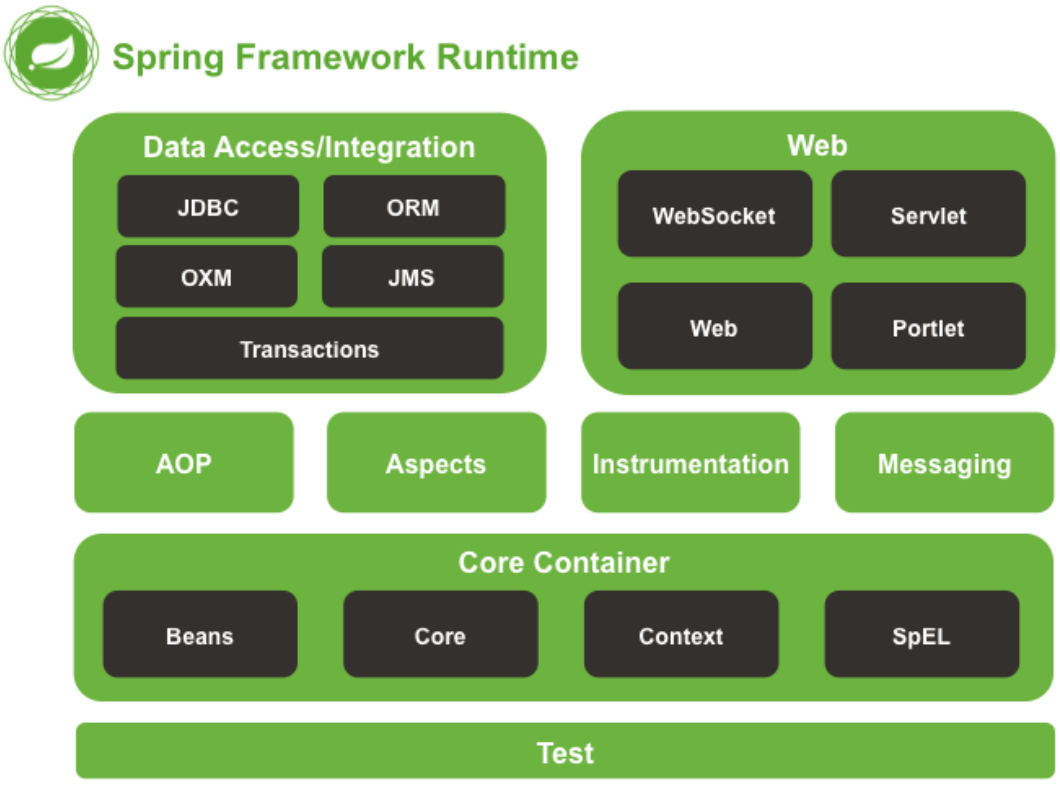
2.3 AJAX

AJAX on JavaScriptiin pohjautuva tekniikka, jota hyödynnetään lähinnä datan päivittämiseen eikä verkkosivua tarvitse ladata uudestaan. AJAX:lla tehdään kutsu (liite 1), jolla viedään JQuery:llä serialisoitu JavaScript-olio url-osoitteeseen ja alustetaan yhteys tulevaa kutsua varten. Kutsun suorittamisen jälkeen kutsuttu metodi palauttaa vastauksen JSON-muodossa. JSON-muotoinen vastaus voidaan päivittää verkkosivulle jQuery-tekniikkaa hyödyntäen (Asleson & Schutta 2007,16-17).

2.4 Java

Java on objekti-orientoitunut ohjelmointikieli, joka on suosituin lähestymistapa web-ohjelmointiin. Javaan on saatavilla useita alustoja, kuten Spring, joka helpottaa ohjelmointia suunnattomasti (Bell & Parr 2005).

Kuva 1. Kokonaiskuva Spring Frameworkista



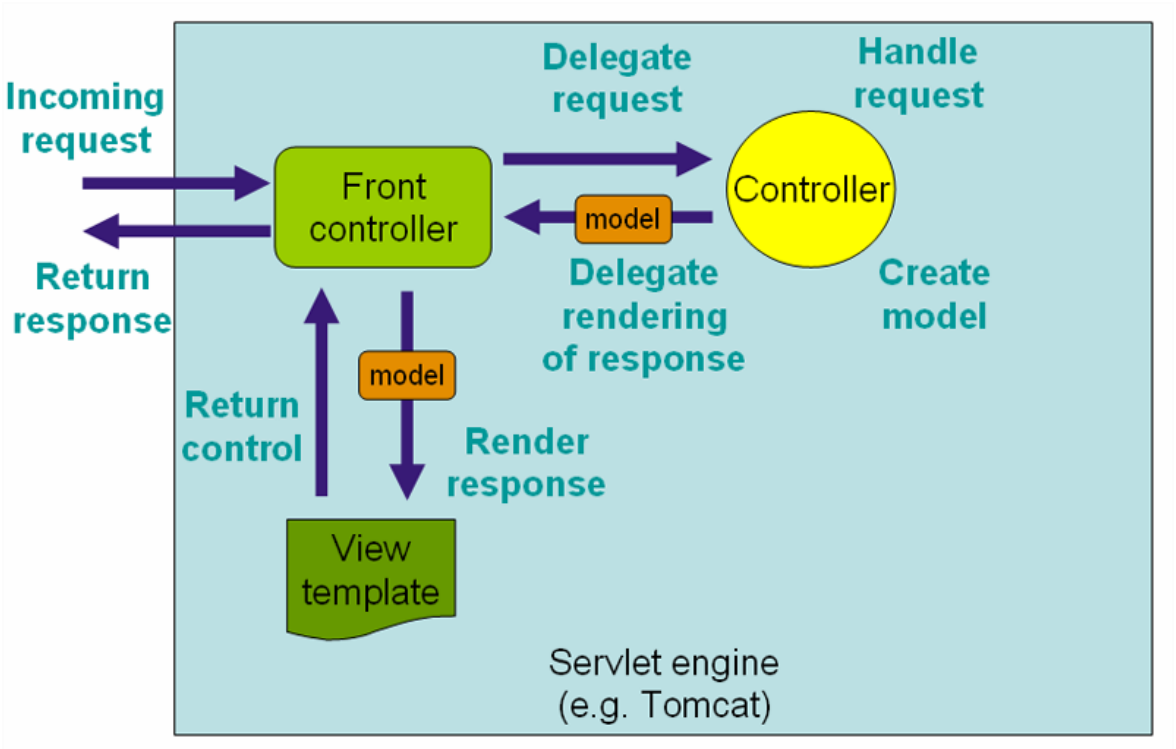
(Spring 2014c)

Kuvassa (kuva 1) kuvattu Core Container-osio poistaa tarpeen hyödyntää kertakäyttöisiä muuttujia, jolloin niitä ei tarvitse ottaa millään tavalla huomioon ohjelmoitaessa vaan muuttujat tulevat suoraan bean-luokasta. Näin ollen toistuvaa koodia ei tullut niin paljon ja tehty työ on suunnattomasti pienempi.

Kuvassa (kuva 1) kuvattu Data Access/Integration-osio mahdollistaa transaktiot. Transaktioiden avulla tehdään muutoksia tietokantaan turvallisesti eikä tietokantaan pääse vääränlaista dataa. Projektin kannalta datan pysyminen eheänä oli ehdottoman tärkeää.

Kuvassa (kuva 1) Web layer-osio mahdollistaa objekti-pohjaiset pyynnöt, joka tekee sovelluksesta todella joustavan. Tuodut objektit myös käsitellään pyyntöjen yhteydessä, jolloin vääränlaiset objektit jätetään kokonaan ulkopuolelle. Esimerkiksi AJAX-kutsut kulkevat web layerin kautta hyödyntäen DispatcherServlettiä, joka on toiselta nimeltää Front controller. Kuvassa (kuva 2) kuvataan DispatcherServletin kommunikointia ja sen suurta roolia web-sovelluksessa.

Kuva 2. Front controllerin toimintamalli



(Spring 2014b).

2.5 Hash

Salasanan tarkoituksena on pysyä salassa, jolloin kenenkään muun kuin sen luojan ei tulisi sitä tietää. Salasana pysyy parhaiten salassa, kun sitä ei tallenneta missään vaiheessa

palveluun vaan siitä luodaan hash, joka tallennetaan palveluun. Hash:n avulla varmistetaan tarvittaessa salasanan oikeellisuus siten, että annetusta salasanasta luodaan uusi hash, jota verrataan palveluun tallennettuun hash:iin. Hash-tekniikan käyttö estää tilanteen, jossa palveluun murtaudutaan ja tietokannasta viedään kaikki salasanat. Kun salasana tallennetaan muotoon, joka ei vastaa alkuperäistä muotoa lainkaan on mahdotonta päätellä alkuperäinen salasana, joka vähintäänkin hidastaa varastettujen salasanojen käyttöä. Hash-muotoinen salasana ei mahdollista salasanan palauttamista millään tavalla vaan käyttäjälle joudutaan luomaan uusi väliaikainen salasana. Aina voidaan myös pohtia onko hash-menetelmä tarpeeksi tehokas esimerkiksi kymmenen vuoden kuluttua vai joudutaanko silloin hyödyntämään tehokkaampaa menetelmää, jolloin salasanan vaihto saattaisi olla hyvinkin hankalaa, koska muutosprosessi saattaisi tehdä palvelun alttiiksi hyökkäyksille (Gates, 301).

Hash tuottaa erilaisen tulosteen, jokaisesta datan osasta, mutta täysin identtinen data tuottaa tismalleen samanlaisen tulosteen kuin alkuperäinen tuloste. Datan muuttuessa myös tuloste muuttuu. Hash-tekniikasta on saatavilla useita eri menetelmiä, kuten SHA-256 ja menetelmästä riippuen hash:n pituus vaihtelee (Gates, 301).

2.6 Tunnelointi

PuTTY on asiakaslähtöinen ohjelma tietoverkon protokolille. Näiden protokolien avulla voidaan muodostaa etäyhteyksiä verkon kautta toiseen tietokoneeseen. PuTTY:n avulla on mahdollista tunneloida eli määrittää portin, jonka kautta yhteys muodostetaan. Tietokantaan yhdistäessä käytetään usein porttia 3306 (PuTTY 2014).

2.7 Tietokanta

Sovelluksessa hyödynnetään MySQL-tietokantaa, joten projektikin on käytännöllistä toteuttaa sillä. MySQL on yksi tehokkaimmista tietokantajärjestelmistä, jonka ominaisuuksiin kuuluu:

- Relaationaallisuus
- Molemminpuolinen arkkitehtuuri asiakas/serveri
- SQL yhteensopivuus
- Käyttöliittymä
- Kokonais-tekstihaku
- Tiedon kopiointi
- Transaktio-tuki
- Viiteavaimet
- Tuki usealla ohjelmointikielelle, kuten Javalle
- Tuki usealla käyttöjärjestelmälle
- Nopeus

(Kofler, 6-8).

Tietokannassa yleisesti hyödynnettyjä lauseita ovat:

- DELETE
- INSERT
- SELECT
- UPDATE

(MySQL 2014b).

SELECT-lauseeseen voidaan lisätä LIMIT rajoittamaan palautettavan tiedon määrää. Limit-vaihtoehdolla dataa voidaan rajoittaa näin SELECT * FROM user WHERE id=id LIMIT 1,10, jossa toinen täysluku määrittää kuinka monta riviä haetaan ja ensimmäinen luku määrittää monesko sivu on kyseessä (MySQL 2014a).

Välillä on tarpeellista tietää myös kuinka monta sivua haettua tietoa on yhteensä saatavilla. SQL_CALC_FOUND_ROWS osana SELECT-lausetta pakottaa MySQL:n laskemaan kuinka monta riviä on yhteensä haettavissa. Tiedon voi noutaa SELECT-lauseen jälkeen tekemällä SELECT-lauseen, jossa on FOUND_ROWS() mukana (MySQL 2014a).

2.8 Hyödynnetyt menetelmät

Tekniikoita on useita ja niiden mukana tulee useita menetelmiä. Menetelmien merkittävyyttä on hyvä arvioida, jotta saadaan kartoitettua projektin kannalta tarpeelliset menetelmät. Menetelmiä arvioidaan seuraavasti:

- Suuri: ilman tätä menetelmää projektia ei olisi voitu toteuttaa.
- Keskisuuri: datan eheys tai tiedon turvallisuus saattaa olla uhattuna ilman tätä menetelmää ja menetelmää.
- Pieni: menetelmä on hyödyllinen, mutta ei ratkaiseva datan eheyden tai tiedon turvallisuuden kannalta.

2.8.1 Validate

JQuery-tekniikalla validointi mahdollistaa tiedon tarkastamisen hyödyntäen pelkkää yhtä riviä ja asetettua validointi sääntöjä. $("#commentForm").validate(); rivin avulla haetaan dokumentista id nimeltä commentForm, jolle pyydetään validointia. Validoinnin yhdessä sulkujen sisälle tai input-kenttään määritetään säännöt jokaiselle lomakkeen kohdalle. Esimerkiksi validate({rules : {id : {required : true} } }) määrittää id-kentälle arvon required, jolloin tämä kenttä vaaditaan määritellysti. Kentän arvolle voidaan määrittää myös halutut virheilmoitukset, joiden mukaan käyttäjä pystyy korjaamaan syötteen. Kentälle voidaan määrittää esimerkiksi seuraavat arvot:

- required: kenttä vaaditaan tai sitä ei vaadita, jolloin kentän arvo on joko true tai false.
- minlength: kentän tulee olla vähintäänkin tämän pituinen, jolloin kentän arvo on kokonaisluku.

- maxlength: kentän arvo saa olla maksimissaan tämän pituinen, jolloin kentän arvo on kokonaisluku.
- equalTo: kentän tulee täsmätä toiseen kenttään, jolloin kentän arvoksi määritetään toinen kenttä. Tätä menetelmää hyödynnetään usein salasanan vahvistamisessa.

(Validate 2014).

Menetelmän avulla voidaan kertoa käyttäjälle, että tämän syöttämä data on virheellistä ilman, että tietoa lähetetään eteenpäin. Resursseja säästetään, kun tietoa ei lähetetä heti eteenpäin vaan se käsitellään ensin. Käyttäjä saa myös reaaliaikaisesti tietoa syötetyn datan oikeellisuudesta. Valitettavasti menetelmän voi kiertää ottamalla skriptikielet pois käytöstä, joten menetelmän merkittävyys jää pieneksi.

2.8.2 JSON-stringify

JSON-stringify-menetelmä muuntaa JavaScriptin dataa JSON-muotoiseksi tekstiksi, jolloin haluttu tieto voidaan välittää eteenpäin esimerkiksi AJAX-kutsussa. Serializella muunnetaan kokonainen lomake, mutta JSON-stringify vie vain yhden osan dataa (JSON 2014b).

Ilman tätä menetelmää dataa ei voisi lähettää osissa. Tietoa tulee pystyä kumminkin lähettämään eteenpäin pienemmässäkin koossa, mutta sen pois jättäminen ei vaaranna turvallisuutta tai tiedon eheyttä, joten menetelmän merkittävyys on keskisuuri.

2.8.3 Serialize

JQuery:n serialize-menetelmä muuntaa lomakkeen tiedot merkkijonoksi, jolloin tieto voidaan lähettää esimerkiksi AJAX-kutsussa eteenpäin käsiteltäväksi. $("form").serialize(); dokumentista haetaan form-lomake, joka sitten muunnetaan haluttuun muotoon (jQuery 2014).

Ilman tätä menetelmää tieto pitäisi välittää karkeasti JSON-stringify-menetelmän avulla, jolloin työtä tulisi paljon enemmän. Menetelmä helpottaa sovelluskehitystä huomattavasti, mutta sen pois jättäminen ei vaaranna turvallisuutta tai tiedon eheyttä, joten sen merkittävyys on keskisuuri.

2.8.4 Bean validointi

Spring sisältää useita valmiita validointi-menetelmiä, joilla voidaan rajoittaa datan pääsyä eteenpäin. Metodiin määritelty bean-validointi kertoo sovellukselle, että bean:iin asetetut

rajoitteet tulee tarkistaa, joiden mukaan data joko hylätään tai hyväksytään. Bean:iin määritetään rajoite lisäämällä @rajoitteennimi vaaditun arvon yläpuolelle. Bean:iin voi määrittää esimerkiksi nämä rajoitteet:

- NotNull: arvo ei saa olla null-arvo.
- Size: arvon pituuteen voi määritellä max-arvon, jolla määritetään arvon maksimipituus.
- Min: Määritetään minipituus arvolle.

Lisäksi mukaan voi kustomoida omia rajoitteita, joka mahdollistaa todella joustavan tiedon validoinnin (Spring 2014e).

Menetelmän merkittävyys on suuri, koska tätä validointia asiakas ei kykene kiertämään ja siihen on saatavilla todella paljon valmiita rajoitteita ja rajoitteita voi luoda myös itse. Vääränlainen data on myös helppo estää tällä menetelmällä.

2.8.5 SHA-256

SHA-256-menetelmä tulostaa 64 merkkiä pitkän hash:n joka on tällä hetkellä yksi seitsemästä hyväksytystä algoritmista (Information Technology Laboratory 2014a; Information Technology Laboratory 2014b).

Menetelmän merkittävyys on suuri, koska tämä menetelmä on yksi hyväksytystä algoritmista näin ollen riittävä menetelmä hash:n luontiin. Lisäksi tämä menetelmä vaaditaan maksuprosessiin.

2.8.6 HMAC

HMAC perustuu MAC-menetelmään, jolla varmistetaan tiedon lähde ja sen eheys, mutta näiden lisäksi data välitetään myöskin hash-muodossa. Datan oikeellisuuden varmistamiseen käytetään salaista avainta, joka on vain viestin lähettäjän ja vastaanottajan hallussa. MAC luodaan välitettävästä datasta ja salaisesta avaimesta ja se lähetetään tiedon mukana, jotta vastaanottaja voi luoda oman MAC-arvon ja verrata sitä vastaanotettuun MAC-arvoon. kun MAC-arvot täsmäävät voidaan tieto lukea eheäksi (Information Technology Laboratory 2014b).

HMAC-menetelmä on merkittävyydeltään suuri, koska muokatun datan huomaa välittömästi, jolloin tietoon ei voi enää luottaa. Lisäksi HMAC on helposti päivitettävissä uudempaan versioon, koska uudempien hash-menetelmien myötä myös HMAC:n turvallisuus vahvistuu.

2.8.7 Transaktiot

Spring sisältää myös transaktio-tuen, joka tarjoaa seuraavat hyödyt:

- Yhtenäinen ohjelmointimalli usealle transaktio-API:lle.

- Transaktion voi ottaa käyttöön pelkällä annotaatio-merkinnällä.
- Yksinkertaisempi kuin muut vaihtoehdot.

Transaktioita voi hyödyntää globaalisti tai lokaalisti, mutta molemmissa on hyvät ja huonot puolensa. Globaalit transaktiot mahdollistavat työskentelyn useiden eri transaktio lähteiden parissa. Lokaalit transaktiot sen sijaan ovat helpompia käyttää, mutta eivät mahdollista työskentelyä usean eri transaktio lähteen parissa. Useat sovellukset kumminkin hyödyntävät vain yhtä transaktio lähdettä, jolloin globaaleille transaktioille ei ole tarvetta (Spring 2014d).

Transaktion voi ottaa esimerkiksi käyttöön näin @Transactional(readOnly = false, propagation = Propagation.REQUIRES_NEW). Annotaatiolle voi määritellä seuraavat asetukset:

- Value
- Propagation
- Isolation
- readOnly
- Timeout
- rollbackFor.
- rollbackForClassName
- noRollBackFor
- noRollBackForClassName

(Spring 2014d).

Menetelmän merkittävyys on suuri, koska menetelmää ei tarvitse erikseen lisätä sovellukseen vaan se tulee Spring:n mukana. Transaktion käyttöönotto on todella yksinkertaista ja jokaiselle metodille voi helposti lisätä omat asetukset transaktioille, esimerkiksi SELECT-lauseesta voi tehdä pelkästään luettavan, jolloin tietoon ei voi tehdä muutoksia. Lisäksi tietoa voidaan palauttaa todella joustavasti jos tiedon muokkaaminen epäonnistuu ja estää tiedon käyttö jos se vaarantaa tiedon eheyden.

2.8.8 Testaus

Aikataulun ollessa tiukka on tavoitteena löytämään niin monta bugia kuin mahdollista. Välillä on kumminkin hyvä antaa toisen henkilön käydä läpi tekemäsi testit, jotta voidaan löytää myös huomaamatta jääneet bugit (Patton 2005, 254). Testaus on tärkeää koko projektin kannalta.

2.8.9 Haastattelu

Projektin aikana haastatellaan projektin toimeksiantajaa, jotta saadaan kirjoitettua tarkat käyttötapaukset, joiden avulla määritellään ominaisuuksien toteutus. Haastattelun avulla saadaan myös opinnäytetyöhön mukaan myös projektin toimeksiantajan näkökulma siitä

kuinka projekti sujui. Projektin toimeksiantajalle lähetetään kysymyksiä sähköpostitse, kun projekti on toteutettu.

2.9 Vaatimusmäärittely

Sovellusta pääsee käyttämään vain rekisteröitynyt käyttäjä eivätkä tiedot lukuun ottamatta kirjautumissivua ole kaikkien luettavissa. Poikkeuksena ovat admin-käyttäjälle suunnatut hallintatyökalut, joihin pääsee käsiksi vain palvelun hallinnoija.

Sovellukseen määritetyt käyttäjäroolit:

- Admin
- Asiakkaan admin
- Rekisteröitynyt käyttäjä

Rekisteröitynyt käyttäjä on asiakkaan admin-käyttäjän luoma käyttäjä, jolla on oikeudet asiakkaalle määritellylle sivulle. Asiakkaan admin-käyttäjällä on samat oikeudet kuin rekisteröityneellä käyttäjällä, mutta sen lisäksi se voi hallinnoida oman yrityksensä rekisteröityneitä käyttäjiä. Admin-käyttäjällä on samat oikeudet kuin rekisteröityneellä käyttäjällä, mutta samalla pääsy päähallintaan, jossa voi hallinnoida asiakkaita.

Vaatimusmäärittely toteutetaan hyödyntäen sovellukseen määriteltyjä käyttäjärooleja, joiden mukaan käyttötapaukset määritetään. Käyttötapauksista pyritään saamaan mahdollisimman tarkkoja, joten apuna käytetään haastattelua.

3 Projekti

3.1 Käytettävät resurssit

Projektin toimeksiantaja tarjoaa tietokannan projektin toteuttamisen ajaksi, jotta erillistä testaamiseen käytettävää materiaalia ei tarvitse luoda. Projektin toimeksiantaja on myös mukana testaamassa uusia ominaisuuksia, jotta ominaisuuksista saadaan mahdollisimman aukottomia.

3.2 Riskit

Projektin toimeksiantajan liiketoiminnan kannalta suurena riskinä voidaan nähdä projektipäällikön sairastuminen, jolloin projekti ei etene lainkaan tai etenee todella hitaasti. Sairastuminen voi kestää useita viikkoja ja projekti saattaa lykkääntyä todella paljon tai se voidaan kokonaan lopettaa. Riskin merkittävyys on suuri eikä sitä voi juuri millään tavalla ennakoida.

Liiketoiminnan kannalta merkittävän riskinä voidaan pitää myös haluttujen ominaisuuksien valmistumista väärällä prioriteetilla, jolloin sovelluksen markkinointia ei voida aloittaa tavoitellusti. Riskin merkittävyys on suuri, mutta se voi aiheuta vain jos kommunikointi ei toimi projektipäällikön ja projektin toimeksiantajan kanssa.

Liiketoiminnan kannalta myös turvallisuus nähdään suurena riskinä maksuliikenteessä, koska virhe voi johtaa suuriin taloudellisiin menetyksiin. Riskin merkittävyys on suuri, mutta se voi aiheutua voin jos vaatimusmäärittelyä ei tehdä kunnolla.

Palvelimen vioittuminen on aina hidaste sovelluskehitykselle, mutta hyvin talteen otetut varmuuskopiot pienentävät riskiä, vaikka kehitysympäristön palauttaminen saattaakin viedä jonkin verran aikaa. Riskin merkittävyys on pieni, koska varmuuskopiointi on kunnossa ja ajan menetys ei ole kovin suuri.

3.3 Vaatimusmääritys

3.3.1 Asiakkaiden lisääminen

Asiakkaalle tulee pystyä luomaan tili, jolle luodaan luontivaiheessa pääkäyttäjä. Pääkäyttäjälle määritetään myös salasana ja sen tulee täsmätä, kun se vahvistetaan toisessa kentässä. Tilin luontivaiheessa pyydetään seuraavat tiedot:

- Y-Tunnus
- Yrityksen nimi

- Puhelinnumero
- Sähköposti
- Katuosoite
- Postinumero
- Postitoimipaikka
- Käyttäjien määrä
- Käyttäjätunnus
- Salasana
- Salasana uudestaan

Kenttien arvojen tulee vastata seuraavia kriteerejä:

- Y-tunnus vaaditaan ja sen tulee olla vähintään 9-merkkiä pitkä.
- Yrityksen nimi vaaditaan, sen tulee olla vähintään 3-merkkiä pitkä ja se saa olla maksimissaan 200-merkkiä pitkä.
- Puhelinnumeroa ei vaadita, mutta sen tulee olla vähintään 3-merkkiä pitkä ja maksimissaan 50-merkkiä pitkä.
- Sähköposti vaaditaan ja sen tulee olla vähintään 3-merkkiä pitkä ja maksimissaan 50-merkkiä pitkä.
- Katuosoite vaaditaan ja sen tulee olla vähintään 3-merkkiä pitkä ja maksimissaan 50-merkkiä pitkä.
- Postinumero vaaditaan ja sen tulee olla vähintään 5-merkkiä pitkä ja maksimissaan 5-merkkiä pitkä.
- Postitoimipaikka vaaditaan ja sen tulee olla vähintään 3-merkkiä pitkä ja maksimissaan 50-merkkiä pitkä.
- Maksimikäyttäjiä ei vaadita, koska siihen on määritetty jo valmiiksi oletusarvo.
- Käyttäjätunnus vaaditaan ja sen tulee olla vähintään 3-merkkiä pitkä ja maksimissaan 50-merkkiä pitkä.
- Salasana vaaditaan ja sen tulee olla vähintään 8-merkkiä pitkä ja maksimissaan 50-merkkiä pitkä.
- Salasanan toistokenttä ja sen tulee vastata edellistä salasana kenttää.

Projektin toimeksiantaja tarvitsee ominaisuuden, jotta asiakkaat voivat käyttää heidän palveluaan. Ominaisuuden tarpeellisuuden vuoksi prioriteetti on suuri. Ominaisuus hyväksytään, kun asiakkaalle luodaan tili ja pääkäyttäjä sekä annetut validointi kriteerit täytetään.

Käyttötapauksen nimi	Asiakkaan lisääminen
Toimija	Admin
Esiehto	Käyttäjätili on luotu ja sillä on tarvittavat oikeudet
Lopputulos	Asiakas ja pääkäyttäjän tunnus lisätään tietokantaan ja niille asetetaan tarvittavat oikeudet.
Käyttötiheys	20-50/24 tuntia

1. Käyttötapaus näyttää päähallinnan toimijalle.
2. Toimija painaa ”Lisää”-nappia.

3. Toimija antaa pakolliset tiedot eli Y-tunnuksen, yrityksen nimen, puhelinnumeron, sähköpostin, katuosoitteen, postinumeron, postitoimipaikan ja valitsee käyttäjien määrän.
4. Toimija antaa pääkäyttäjälle käyttäjätunnuksen ja salasanan sekä vahvistaa salasanan.
5. Toimija painaa toista ”Lisää”-nappia.
6. Käyttötapaus määrittää lisätylle käyttäjälle asiakas admin-käyttäjän oikeudet sekä lisää asiakkaan tietokantaan.
7. Käyttötapaus sulkee ikkunan.
8. Käyttötapaus päättyy.

Varitaatiot
Toimija keskeyttää asiakkaan luomisen ja käyttötapaus päättyy. Toimija saa virheilmoituksen. Käyttötapaus ei pääty vaan kehottaa toimijaa tekemään korjauksia.
Poikkeukset
Toimija saa virheilmoituksen. Käyttötapaus päättyy.

3.3.2 Asiakkaiden listaaminen

Asiakkaita tulee pystyä listaamaan, jotta valittua asiakasta pystyy hallinnoimaan, jonka vuoksi sen prioriteetti on suuri. Ominaisuus tarvitaan juurikin sen tärkeyden vuoksi. Ominaisuus hyväksytään, kun kaikki asiakkaat listautuvat sivulle ja riviltä löytyy ”Hallinnoi”-nappi.

Asiakkaista halutaan myös listata seuraavat tiedot:

- Y-tunnus
- Yrityksen nimi
- Sähköposti
- Puhelin
- Pääkäyttäjä
- Kuinka monta tili on käytössä
- Hallinnoi-nappi

Käyttötapauksen nimi	Asiakkaiden listaaminen
Toimija	Admin
Esiehto	Käyttäjätili on luotu ja sillä on tarvittavat oikeudet
Lopputulos	Asiakas näkee listan asiakkaista.
Käyttötiheys	50-100/24 tuntia

1. Käyttötapaus näyttää päähallinnan toimijalle.
2. Käyttötapaus listaa asiakkaat.

3. Käyttötapaus päättyy.

Varitaatiot
Varitaatioita ei ole.
Poikkeukset
Toimija saa virheilmoituksen. Käyttötapaus päättyy.

3.3.3 Asiakkaan muokkaaminen

Asiakkaita pitää pystyä tarpeen tullessa muokkaamaan. Projektin toimeksiantaja tarvitsee ominaisuuden laskutusosoitteen päivittämisen vuoksi, joka tekee sen prioriteetista suuren. Laskutusosoite takaa sen, että lasku menee oikeaan osoitteeseen. Ominaisuus hyväksytään, kun seuraavien kenttien tietoja pystyy muokkaamaan ja kentät täyttävät vaaditut kriteerit:

- Yrityksen nimi
- Puhelinnumero
- Sähköposti
- Katuosoite
- Postinumero
- Postitoimipaikka
- Maksimikäyttäjien määrä

Kenttien arvojen tulee vastata seuraavia kriteerejä:

- Yrityksen nimi vaaditaan, sen tulee olla vähintään 3-merkkiä pitkä ja se saa olla maksimissaan 200-merkkiä pitkä.
- Puhelinnumeroa ei vaadita, mutta sen tulee olla vähintään 3-merkkiä pitkä ja maksimissaan 50-merkkiä pitkä.
- Sähköposti vaaditaan ja sen tulee olla vähintään 3-merkkiä pitkä ja maksimissaan 50-merkkiä pitkä.
- Katuosoite vaaditaan ja sen tulee olla vähintään 3-merkkiä pitkä ja maksimissaan 50-merkkiä pitkä.
- Postinumero vaaditaan ja sen tulee olla vähintään 5-merkkiä pitkä ja maksimissaan 5-merkkiä pitkä.
- Postitoimipaikka vaaditaan ja sen tulee olla vähintään 3-merkkiä pitkä ja maksimissaan 50-merkkiä pitkä.
- Maksimikäyttäjiä ei vaadita, koska siihen on määritetty jo valmiiksi oletusarvo.

Käyttötapauksen nimi	Asiakkaan muokkaaminen
Toimija	Admin
Esiehto	Käyttäjätili on luotu ja sillä on tarvittavat oikeudet sekä toimija on valinnut asiakkaan, jota hän haluaa muokata.
Lopputulos	Asiakkaan tiedot päivitetään tietokantaan.

Käyttötiheys	1-2/24 tuntia

1. Käyttötapaus näyttää päähallinnan toimijalle.
2. Toimija painaa ”Hallinnoi”-nappia valitsemansa asiakkaan kohdalla.
3. Toimija muokkaa haluamaansa kenttää/kenttiä.
4. Toimija painaa ”Tallenna”-nappia.
5. Käyttötapaus antaa ilmoituksen ”Muokkaus onnistui”
6. Käyttötapaus päättyy.

Varitaatiot
Toimija keskeyttää asiakkaan muokkaamisen ja käyttötapaus päättyy. Toimija saa virheilmoituksen. Käyttötapaus ei pääty vaan kehottaa toimijaa tekemään korjauksia.
Poikkeukset
Toimija saa virheilmoituksen. Käyttötapaus päättyy.

3.3.4 Asiakkaan tilin ja käyttäjätilien jäädyttäminen

Tarpeen tullen asiakkaan tili ja käyttäjätilit tulee pystyä jäädyttämään. Projektin toimeksiantaja tarvitsee ominaisuuden, jotta tilauksen voi päättää eikä jatku ikuisesti ja se tekee sen prioriteetista suuren. Ominaisuus hyväksytään, kun tilin tilaa pystyy muokkaamaan.

Käyttötapauksen nimi	Asiakkaan tilin ja käyttäjätilien jäädyttäminen
Toimija	Admin
Esiehto	Käyttäjätili on luotu ja sillä on tarvittavat oikeudet sekä tiedetään jäädytettävä asiakastili.
Lopputulos	Asiakkaan tili ja käyttäjätilit jäädytetään.
Käyttötiheys	1-5/24 tuntia

1. Käyttötapaus näyttää päähallinnan toimijalle.
2. Toimija painaa ”Hallinnoi”-nappia valitsemansa asiakkaan kohdalla.
3. Toimija vaihtaa tilan aktiivisesta suljettuun.
4. Toimija painaa ”Tallenna”-nappia.
5. Käyttötapaus antaa ilmoituksen ”Muokkaus onnistui”
6. Käyttötapaus päättyy.

Varitaatiot
Toimija keskeyttää tilien jäädyttämisen ja käyttötapaus päättyy.

Poikkeukset
Toimija saa virheilmoituksen. Käyttötapaus päättyy.

3.3.5 Asiakkaan käyttäjien listaaminen

Asiakkaiden käyttäjiä tulee pystyä listaamaan, jotta valittua käyttäjää pystyy hallinnoimaan. Käyttäjän hallinnointi ei ole tärkeä muihin ominaisuuksiin verrattuna, joten sen prioriteetti on keskisuuri. Ominaisuus hyväksytään, kun ”Käyttäjät”-välilehti listaa asiakkaan käyttäjät jättäen ulkopuolelle pääkäyttäjän.

Käyttötapauksen nimi	Asiakkaan käyttäjien listaaminen
Toimija	Admin
Esiehto	Käyttäjätili on luotu ja sillä on tarvittavat oikeudet sekä hallinnoitava asiakas on valittu.
Lopputulos	Asiakkaan käyttäjät listataan.
Käyttötiheys	1-2/24 tuntia

1. Käyttötapaus näyttää päähallinnan toimijalle.
2. Toimija painaa ”Hallinnoi”-nappia valitsemansa asiakkaan kohdalla.
3. Toimija valitsee ”Käyttäjät”-välilehden.
4. Käyttötapaus listaa käyttäjät.
5. Käyttötapaus päättyy.

Varitaatiot
Asiakkaalla ei ole kuin yksi tunnus, jolloin käyttäjiä ei erikseen listata.
Poikkeukset
Toimija saa virheilmoituksen. Käyttötapaus päättyy.

3.3.6 Asiakkaan käyttäjien määrän muokkaaminen

Asiakkaan käyttäjien määrää pitää pystyä tarpeen tullessa muokkaamaan, jotta asiakas voi tilata käyttöönsä lisää käyttäjiä. Lisätyt käyttäjät tuovat lisää kassavirtaa, joka tekee ominaisuuden prioriteetista suuren. Ominaisuus hyväksytään, käyttäjien määrää pystytään muokkaamaan.

Käyttötapauksen nimi	Asiakkaan käyttäjien määrän muokkaaminen

Toimija	Admin
Esiehto	Käyttäjätili on luotu ja sillä on tarvittavat oikeudet sekä hallinnoitava asiakas on valittu.
Lopputulos	Asiakkaan käyttäjien määrää muokataan.
Käyttötiheys	1-10/24 tuntia

1. Käyttötapaus näyttää päähallinnan toimijalle.
2. Toimija painaa ”Hallinnoi”-nappia valitsemansa asiakkaan kohdalla.
3. Toimija muokkaa ”käyttäjien määrä”-kenttään valitsemansa käyttäjien määrän.
4. Toimija painaa ”Tallenna”-nappia.
5. Käyttötapaus tallentaa muokatun käyttäjien määrän tietokantaan.
6. Käyttötapaus antaa ilmoituksen ”Muokkaus onnistui”
7. Käyttötapaus päättyy.

Varitaatiot
Toimija keskeyttää käyttäjien määrän muokkaamisen ja käyttötapaus päättyy.
Poikkeukset
Toimija saa virheilmoituksen. Käyttötapaus päättyy.
Toimija ei löydä listasta tarpeeksi suurta lukua, jolloin muokkaus tehdään sähköpostitse.

3.3.7 Asiakkaan päätilin salasanan resetointi

Asiakkaan päätilin salasana tulee pystyä tarpeen tullen resetoimaan. Projektin toimeksiantaja tarvitsee ominaisuuden tilanteeseen, jossa pääkäyttäjä hukkaa salasanansa ja se tulee pystyä vaihtamaan. Tilistä maksetaan, joten sen tulee olla aina saatavilla ja se tekee tämän ominaisuuden prioriteetista suuren.

Käyttötapauksen nimi	Asiakkaan päätilin salasanan resetointi
Toimija	Admin
Esiehto	Käyttäjätili on luotu ja sillä on tarvittavat oikeudet sekä hallinnoitava asiakas on valittu.
Lopputulos	Asiakkaan pääkäyttäjän salasana resetoituu ja asiakas saa uuden salasanan.
Käyttötiheys	0-1/24 tuntia

1. Käyttötapaus näyttää päähallinnan toimijalle.
2. Toimija painaa ”Hallinnoi”-nappia valitsemansa asiakkaan kohdalla.
3. Toimija painaa ”Resetoi”-nappia.
4. Toimija painaa ”Sulje ikkuna”-nappia.

5. Käyttötapaus vaihtaa pääkäyttäjän salasanan ja lähettää sen päätilille määriteltyyn sähköpostiin.
6. Käyttötapaus antaa ilmoituksen ”Resetointi onnistui”
7. Käyttötapaus päättyy.

Varitaatiot
Toimija keskeyttää resetoinnin ja käyttötapaus päättyy.
Poikkeukset
Toimija saa virheilmoituksen. Käyttötapaus päättyy.
Sähköpostiosoite ei ole validi, jolloin sähköposti ei lähde.

3.3.8 Asiakkaan käyttäjätilin oikeuksien muuttaminen

Asiakkaan käyttäjille tulee pystyä antamaan lisää oikeuksia. Asiakkailla on olemassa automaattisesti yksi pääkäyttäjä, mutta pääkäyttäjäksi tulee pystyä asettamaan myös muukin käyttäjä. Käyttötiheys on sen verran pieni, että sen prioriteetti jää pieneksi.

Käyttötapauksen nimi	Asiakkaan käyttäjätilien oikeuksien muuttaminen
Toimija	Admin
Esiehto	Käyttäjätili on luotu ja sillä on tarvittavat oikeudet sekä hallinnoitava asiakas on valittu.
Lopputulos	Asiakkaan oikeudet vaihtuvat.
Käyttötiheys	0-1/24 tuntia

1. Käyttötapaus näyttää päähallinnan toimijalle.
2. Toimija painaa ”Hallinnoi”-nappia valitsemansa asiakkaan kohdalla.
3. Toimija valitsee ”Käyttäjät”-välilehden.
4. Käyttötapaus listaa käyttäjät.
5. Toimija vaihtaa valitsemansa käyttäjän kentän arvoa haluamakseen.
6. Käyttötapaus päättyy.

Varitaatiot
Asiakkaalla ei ole kuin yksi tunnus, jolloin käyttäjiä ei erikseen listata eikä siten voi muokata.
Poikkeukset
Toimija saa virheilmoituksen. Käyttötapaus päättyy.

3.3.9 Verkkosivujen integrointi

Projektin toimeksiantajalla oli valmiit verkkosivut, jotka eivät millään tavalla viitanneet heidän varsinaiseen palveluunsa. Käyttäjän tuli pystyä navigoimaan projektin toimeksiantajan verkkosivun kautta edellä mainittuun palveluun. Verkkosivujen integrointi on vaatimus, mutta sitä tarvitaan lähinnä maksupalvelua varten, jotta potentiaalisella asiakkaalla on mahdollisuus suorittaa maksu.

3.3.10 Maksuprosessi

Asiakkaan tulee pystyä valitsemaan verkkosivuilta tuote. Tuotetta ei ole erikseen määritelty, koska tuotteistus ei ole osana vaatimuksia. Ominaisuuden prioriteetti on suuri yrityksen liiketoiminnan kannalta, mutta sen käyttöönotto vaatii täysin toimivan asiakashallinnan, joten sitä ei ole käytännöllistä tehdä ennen muita vaatimuksia. Tuotteen ollessa valittuna tulee pystyä siirtymään vahvistus-sivulle, jossa näkyy tuotteesta seuraavat tiedot:

- Tuotteen nimi
- Tuotteen hinta
- Tuotteen kokonaishinta
- Sopimuskausi
  Käyttäjien määrä

Asiakkaan tulee pystyä myös rekisteröitymään, joten asiakkaalta pyydetään myös seuraavat tiedot:

- Y-Tunnus
- Yrityksen nimi
- Puhelinnumero
- Sähköposti
- Katuosoite
- Postinumero
- Postitoimipaikka
- Käyttäjien määrä
- Käyttäjätunnus
- Salasana
- Salasana uudestaan

Lisäksi rekisteröitymisessä hyödynnettäville kentille on asetettu seuraavat kriteerit:

- Y-tunnus vaaditaan ja sen tulee olla vähintään 9-merkkiä pitkä.
- Yrityksen nimi vaaditaan, sen tulee olla vähintään 3-merkkiä pitkä ja se saa olla maksimissaan 200-merkkiä pitkä.
- Puhelinnumeroa ei vaadita, mutta sen tulee olla vähintään 3-merkkiä pitkä ja maksimissaan 50-merkkiä pitkä.
- Sähköposti vaaditaan ja sen tulee olla vähintään 3-merkkiä pitkä ja maksimissaan 50-merkkiä pitkä.
- Katuosoite vaaditaan ja sen tulee olla vähintään 3-merkkiä pitkä ja maksimissaan 50-merkkiä pitkä.

- Postinumero vaaditaan ja sen tulee olla vähintään 5-merkkiä pitkä ja maksimissaan 5-merkkiä pitkä.
- Postitoimipaikka vaaditaan ja sen tulee olla vähintään 3-merkkiä pitkä ja maksimissaan 50-merkkiä pitkä.
- Maksimikäyttäjiä ei vaadita, koska siihen on määritetty jo valmiiksi oletusarvo.
- Käyttäjätunnus vaaditaan ja sen tulee olla vähintään 3-merkkiä pitkä ja maksimissaan 50-merkkiä pitkä.
- Salasana vaaditaan ja sen tulee olla vähintään 8-merkkiä pitkä ja maksimissaan 50-merkkiä pitkä.
- Salasanan toistokenttä ja sen tulee vastata edellistä salasana kenttää.

Annettujen tietojen validoinnin jälkeen tulee pystyä maksamaan tuote, mutta ennen sitä käyttötapaus luo asiakkaalle tilin ja pääkäyttäjän. Ominaisuus hyväksytään, kun:

- Tuotteen voi valita
- Vahvistus-sivulla näytetään vaaditut kentät
- Tuotteesta näytetään vaadittavat tiedot
- Vaaditut kentät validoidaan
- Maksusuoritus on turvallinen
- Asiakastili ja pääkäyttäjä luodaan
- Asiakastili ja pääkäyttäjä avataan onnistuneen maksun päätteeksi
- Asiakas ohjataan takaisin projektin toimeksiantajan verkkosivulle

3.4 Yleiset vaatimukset

3.4.1 Virheraportin lähettäminen

Projektin toimeksiantajan mukaan virheraportit tulee välittää jo olemassa olevaan sähköpostiosoitteeseen, jossa niitä olisi helppo tallettaa, lukea ja poistaa. Virheraportin sisällön ei tarvitse sisältää mitään muuta, kuin sovelluksen automaattisesti luoman virheilmoituksen.

3.4.2 Optimointi

Toteutetun ratkaisun tulee olla optimoitu. Optimoitu ratkaisu säästää kuluja, kun haetaan sopivia määriä tietoa. Optimointi tulee ottaa huomioon SQL-lauseissa.

3.4.3 Maksun turvallisuus

Projektin toimeksiantajan mukaan maksuprosessissa käsiteltävän datan tulee liikkua turvallisesti. Data voidaan luokitella turvalliseksi, kun siinä hyödynnetään hash-menetelmää, joka on hyväksytty ja luokiteltu turvalliseksi.

3.4.4 Validointi

Palvelun tuli myös olla käyttäjäystävällinen ja siten myös turvallinen, jotta väärää dataa ei pääse läpi. Validointi toteutettiin front-endissä, back-endissä ja lopuksi tietokannassa.

Front-end validoinnissa hyödynnetään jo aikaisemmin mainittua jQuery-tekniikkaa. Front-end validointi kertoo käyttäjälle reaaliaikaisesti jos tämä kirjoittaa kenttiin vääränlaista dataa. JQuery-validoinnissa jokaiselle input-kentälle asetetaan kriteerit, joiden mukaan data validoidaan ja onnistuneen validoinnin jälkeen dataa lähdetään viemään back-endiin. Front-end validoinnin tehtävä ei ole estää vääränlaisen datan pääsyä tietokantaan asti vaan kertoa käyttäjälle, että tämä on tehnyt virheen.

Back-end validointi on viimeinen paikka validoida välitetty data ennen niiden viemistä tietokantaan. Bean-validoinnissa jokaiselle muuttujalle asetetaan kriteerit, joiden mukaan data validoidaan. Käytännössä back-end validointi ei paljoa eroa front-end validoinnista, mutta front-end validointi on käyttäjän puolella ja siten helposti kierrettävissä. Epäonnistunut validointi palauttaa virhekoodin, jolloin data hylätään eikä sitä viedä tämän pitemmälle.

Tietokantavalidointi on viimeinen paikka tarkistaa datan oikeellisuus. Muuttujille asetetaan tarvittavat kriteerit, joiden mukaan data validoidaan. Virheellinen data hylätään välittömästi jos se ei vastaa sille asetettuja kriteerejä. Virheellinen data palauttaa virhekoodin.

3.5 Työtavat ja työn vaiheistus

Projektiin sisältyvät ominaisuudet ovat melko suoraviivaisia, joten määrittely ei tule olemaan kovin laaja eikä se siten vie paljoa aikaa. Määrittelyn yhteydessä tehdään tarpeelliset dokumentit, jotta ominaisuuksia on helppo kehittää myös jatkossa. Viikon alussa pyritään hoitamaan määrittely ja sitten siirrytään toteuttamaan ominaisuutta, jolle määrittely tehtiin. Joihinkin ominaisuuksiin kuluu viikko, mutta osasta saattaa selvitä alle viikolla, joten ajankäytön kanssa ei pitäisi olla ongelmia. Testaus ajoitetaan loppuviikkoon.

Liikkeelle lähdetään asiakkaiden lisäämisestä rekisteriin ja asiakastilejen hallinnoinnista viikoilla 37-40 (liite 3). Virheraporttien välittämiseen keskitytään viikolla 41(liite 3), front-validointiin keskitytään viikolla 42 (liite 3) ja lopuksi maksujärjestelmä toteutetaan viikoilla 43-45 (liite 3). Projektin toimeksiantajan toivomat prioriteetit perustuvat käynnissä olevaan pilottivaiheeseen ja pian alkavaan markkinointiin.

3.6 Ominaisuuksien toteuttaminen

Ominaisuudet on esitelty työn vaiheistuksessa (liite 3) melko suppeasti, joten niiden purkaminen pienempiin kokonaisuuksiin oli projektin kannalta järkevää. Projektin toimeksiantajalle oli helpompaa selittää työn vaihe, kun oli pieniä kokonaisuuksia, joita näyttää sen sijaan, että yksi suuri kokonaisuus tehtäisiin valmiiksi kerralla. Asiakkaiden hallinnassa ominaisuudet olivat jo alusta alkaen todella hyvin selvillä ja jaettuna hieman pienempiin

kokonaisuuksiin, mutta esimerkiksi maksusuorituksessa tätä ei ollut. Työn vaiheet on jaettu pienempiin osiin Product Backlogissa (liite 4). Projektin toimeksiantajan takaamaan tietokantaa hyödynnettiin ottamalla siihen yhteyttä PuTTY:n tunnelin kautta ja data oli helposti saatavilla koko projektin ajan.

3.6.1 Asiakkaiden hallinta

Asiakkaiden hallintaan liittyvät käyttäjätarinat 1-3, 6 ja 8-9, jotka löytyvät Product Backlogista (liite 4). Asiakkaiden hakeminen on optimoitu lisäämällä SQL-lauseeseen LIMIT, jolloin tietoa haetaan vaaditun määrän mukaan. Asiakkaat haetaan päähallinnan-etusivulle AJAX-kutsulla ja haun jälkeen jokaiselle riville tulostetaan nappi, josta voi hallinnoida asiakasta. Napin tulostamisen yhteydessä tulostetaan myös muut vaaditut kentät. Transaktio säännöksi on asetettu readOnly ja arvoksi true, jotta tieto ei ole kuin luettavissa. Isolation tasoksi riittää READ_COMMITTED, koska kyseessä ei ole niin tärkeä asia kuin maksuliikenne tai maksutiedot ja se nopeuttaa merkittävästi hakua.

Asiakkaan ja pääkäyttäjän lisääminen rekisteriin tapahtuu perinteisellä menetelmällä, jossa ei hyödynnetä AJAX- tai JSON-tekniikkaa, koska tiedot lähetetään yksinkertaisesti lomakkeesta suoraan eteenpäin. JQuery tarkistaa lomakkeen virheellisestä syötteestä, jotta käyttäjä tietää korjata annetun syötteen. Validointisäännöt vastaavat vaatimuksessa asetettuja kriteerejä. Transaktio säännöksi määriteltiin virheestä tapahtuva rollback, jotta tietokantaan ei pääse virheellistä dataa.

Asiakkaan tilin muokkaaminen toteutettiin AJAX-kutsulla, jotta tietojen muokkaus kävisi kivuttomasti eikä käyttäjä joutuisi jatkuvasti latailemaan sivua uudelleen eikä hallinta-näkymä tällöin sulkeudu. JQuery hakee tiedot aluksi lomakkeesta ja JQuery:n serialize muuntaa tiedot oikeaan muotoon, mutta tiedon haun yhteydessä tieto myös validoidaan vaadituilla kriteereillä. Onnistuneen validoinnin jälkeen tiedot lähetetään POST-menetelmällä päivitettäväksi, mutta tiedon haun yhteydessä tieto myös validoidaan vaadituilla kriteereillä. Onnistuneen paluuviestin myötä jQuery-tekniikan avulla tulostetaan sivulle toivottu teksti päivittämisen onnistumisesta. Muokkaamisen yhteydessä pystytään myös muokkaamaan käyttäjien määrää ja tilin tilaa samalla kertaa eikä jokaiselle tarvitse luoda erikseen omaa toimintoa. Transaktio säännöksi määriteltiin virheestä tapahtuva rollback, jotta tietokantaan ei pääse virheellistä dataa.

3.6.2 Käyttäjien hallinta

Käyttäjätarinat 4-5 ja 7 löytyvät Product Backlogista (liite 4) ja ne käsittelevät käyttäjiin liittyviä muutoksia. Käyttäjien hakeminen on optimoitu lisäämällä SQL-lauseeseen LIMIT, jolloin tietoa haetaan vaaditun määrän mukaan. Projektin toimeksiantaja toivoi, että haetaan kymmenen hakutulosta kerralla ja se onnistuu tällä menetelmällä. Käyttäjät haetaan käyttäjät-välilehdelle AJAX-kutsulla ja haun jälkeen jokaiselle riville tulostetaan kenttä, josta voi valita käyttäjän oikeudet. Transaktio säännöksi on asetettu readOnly ja arvoksi true, jotta tieto ei ole kuin luettavissa. Isolation tasoksi riittää READ_COMMITTED, koska kyseessä ei ole niin tärkeä asia kuin maksuliikenne tai maksutiedot ja se nopeuttaa merkittävästi hakua.

Käyttäjälle lisää oikeuksia annettaessa kentästä on ensin haettava käyttäjän id, jonka avulla tehdään AJAX-kutsu. JSON-stringify-menetelmää hyödynnetään tässä tapauksessa, koska tieto haetaan kentästä eikä suoraan lomakkeesta. AJAX-kutsu lähetetään turvallisuuden vuoksi POST-menetelmällä eteenpäin. Samanlainen AJAX-kutsu tehdään myös, kun pääkäyttäjän salasana resetoidaan. Transaktio säännöksi määriteltiin virheestä tapahtuva rollback, jotta tietokantaan ei pääse virheellistä dataa.

3.6.3 Virheraportointi

Käyttäjätarina 12 löytyy Product Backlogista (liite 4), joka käsittelee virheraportointia. Virheraportoinnille on luotu oma metodi, jota kutsutaan yleensä silloin, kun tapahtuu poikkeus sovelluksessa. Metodille ohjataan sovelluksen automaattisesti luoma virhe, joka välitetään sähköpostilla vaadittuun sähköpostiosoitteeseen.

3.6.4 Maksuprosessi

Käyttäjätarinat 16-22 löytyvät Product Backlogista (liite 4) ja ne käsittelevät maksuprosessia, joka itsessään on suuri kokonaisuus, joten se on jaettu useaan osaan helpomman sovelluskehityksen kannalta. Toiminto tapahtuu käyttäjäroolien ulkopuolella eikä ole tavanomainen käyttötapaus, joten turvallisuus tulee huomioida todella tarkkaan. Tilin ja pääkäyttäjän luonnin yhteydessä tili sekä pääkäyttäjä suljetaan, jotta niitä ei pääse käyttämään ennen onnistunutta maksusuoritusta. Maksusuorituksen jälkeen asiakas ohjataan takaisin projektin toimeksiantajan verkkosivulle, jossa asiakas saa joko ilmoituksen maksun onnistumisesta tai maksun epäonnistumisesta. Onnistunut maksu avaa luodun tilin ja pääkäyttäjän ja asiakas pääsee käyttämään palvelua.

Kaikki data maksupalvelun ja sovelluksen välillä kulkee HMAC-muodossa. Luotu HMAC lähetetään maksupalvelulle, joka luo tiedoista maksun. Kun maksu on suoritettu onnistuneesti maksupalvelu palauttaa sovellukselle hash:n JSON-muodossa. JSON-muotoisesta vastauksesta saadaan tiedot maksun onnistumisesta, mutta myös HMAC, joka on luotu maksupalvelussa samoista tiedoista. Sovellus luo vastaanotetuista tiedoista oman HMAC-arvon, jonka tulee täsmätä maksupalvelusta saatuun HMAC-arvoon ja mikäli ne täsmäävät tulkataan maksu onnistuneeksi.

3.6.5 Verkkosivut

Käyttäjätarinat 14 ja 15 löytyvät Product Backlogista (liite 4) ja ne käsittelevät verkkosivujen integrointia ja kontaktointi-lomakkeen korjausta siten, että sähköpostia lähetetään Javalla. Verkkosivut oli alun perin toteutettu toisella ohjelmointikielellä, jolla lähetettiin lomakkeen tiedot eteenpäin sähköpostiosoitteeseen. Verkkosivun sähköpostin lähettämiselle luotiin oma metodi, jolla lähetettiin lomakkeen tiedot vaadittuun sähköpostiosoitteeseen. Verkkosivut ovat osana maksuprosessia, koska maksuprosessi toteutetaan projektin toimeksiantajan verkkosivujen kautta.

3.7 Ominaisuuksien testaaminen

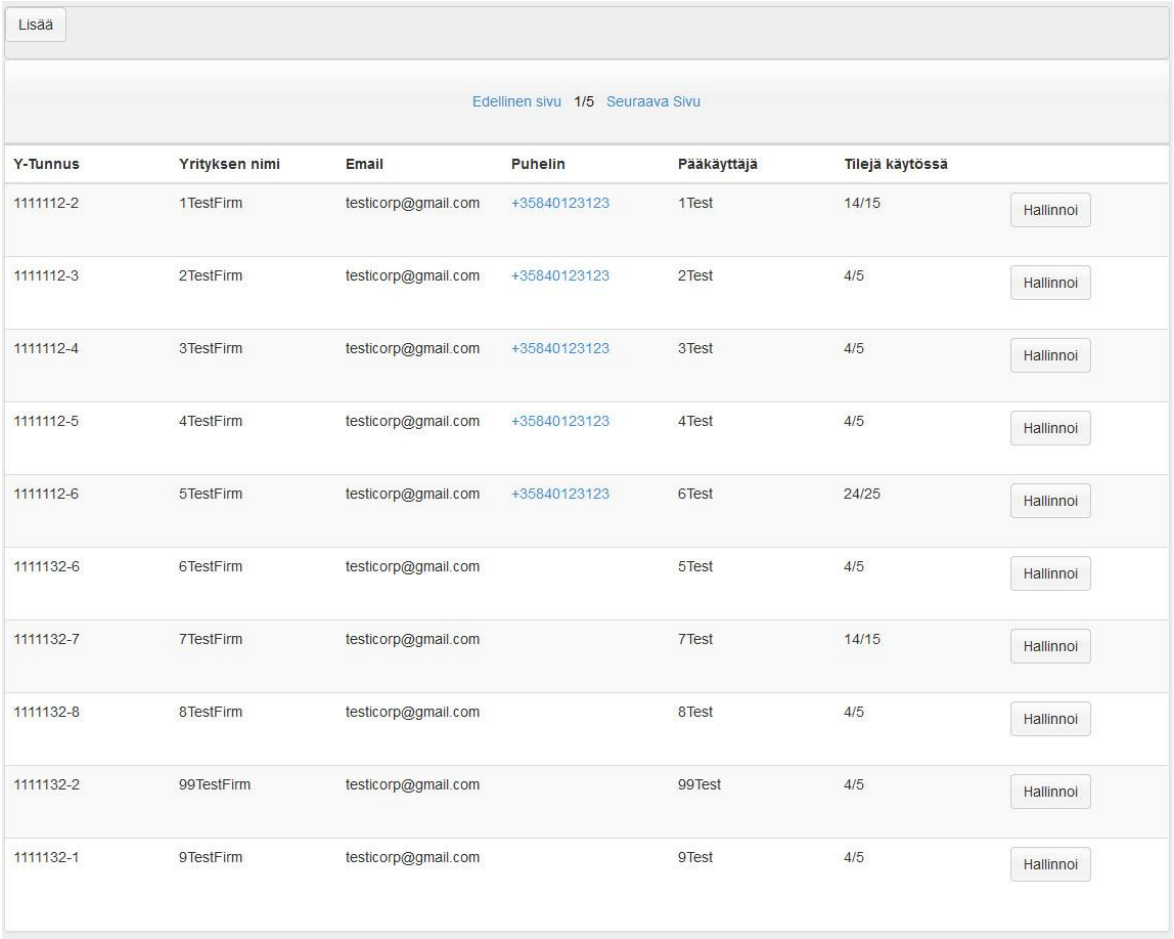
Testaustuloksien (liite 2) kohdat on hyväksytty hyödyntäen testaukseen valittua menetelmää. Projektissa toisena testaajana on toiminut toimeksiantajan testaaja, koska testausta ei ollut mahdollista toteuttaa vain yhden henkilön voimin.

3.8 Produkti

JQuery- ja AJAX-tekniikan avulla ja tarkoin suunniteltujen SQL-lauseiden ansiosta ratkaisu on optimoitu sekä kompakti ja siinä on asiakkaiden hallinta, joka mahdollistaa asiakkaiden lisäämisen ja muokkaamisen käyttäjäystävällisesti, automaattisen virheiden raportoinnin, joka toimii pohjana jatkossakin jokaiselle luodulle metodille ja turvallinen maksuprosessi, jossa hyödynnetään ajan tasalla olevia standardeja.

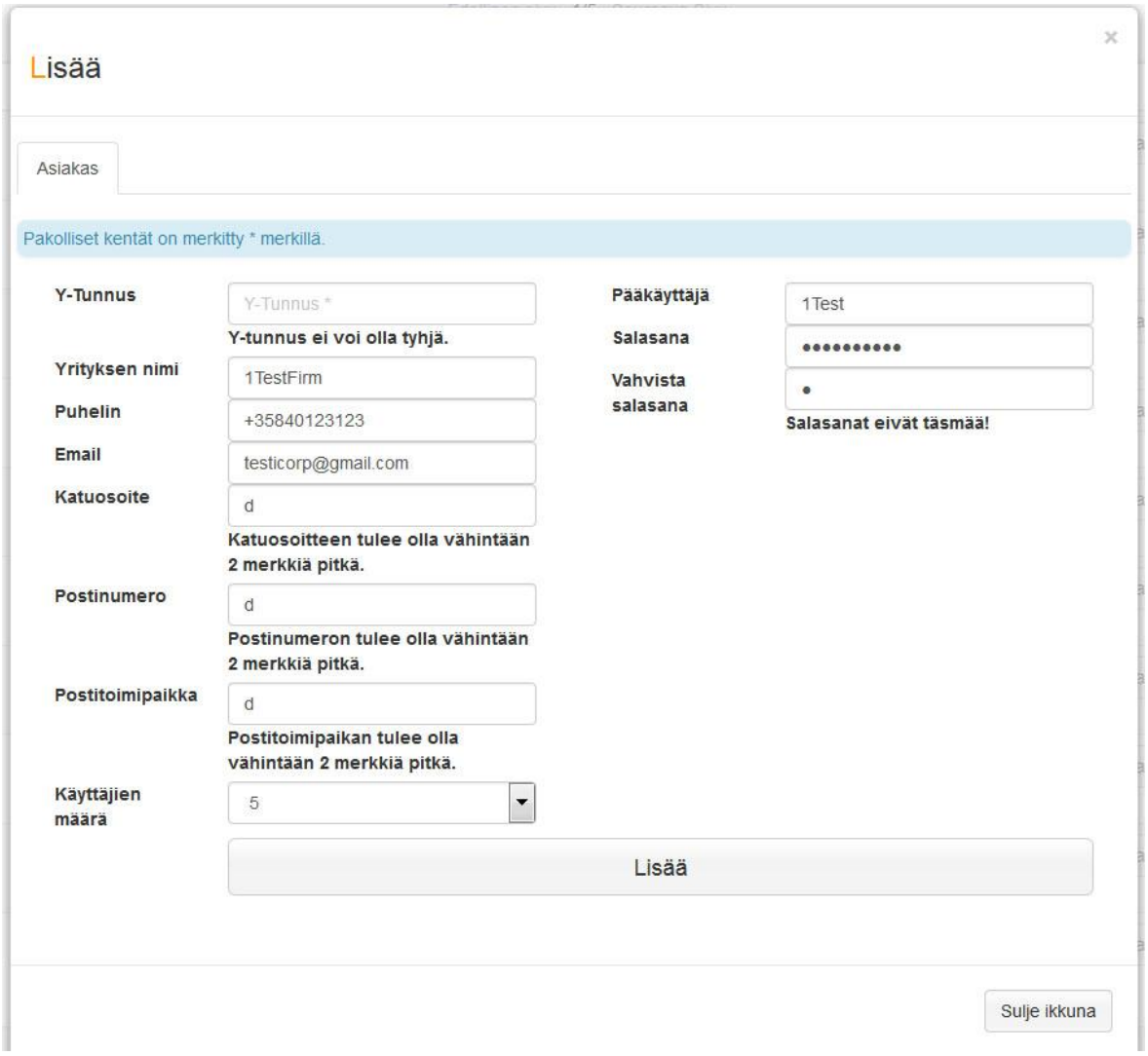
Kuvassa (kuva 3) listataan kymmenen ensimmäiselle sivulle kuuluvaa asiakasta. Asiakkaiden listauksessa haetaan aina kymmenen riviä kerrallaan. Edellinen sivu hakee edelliset kymmenen ja seuraava sivu hakee seuraavat kymmenen riviä. Edellistä sivua painaessa sivunumeron ollessa yksi toimintoa ei voida toteuttaa, koska sivua nolla ei ole olemassa. Seuraavaa sivua ei voida myöskään hakea, kun sivujen maksimimäärä on saavutettu, koska rivejä ei ole enää saatavissa.

Kuva 3. Lista asiakkaista.



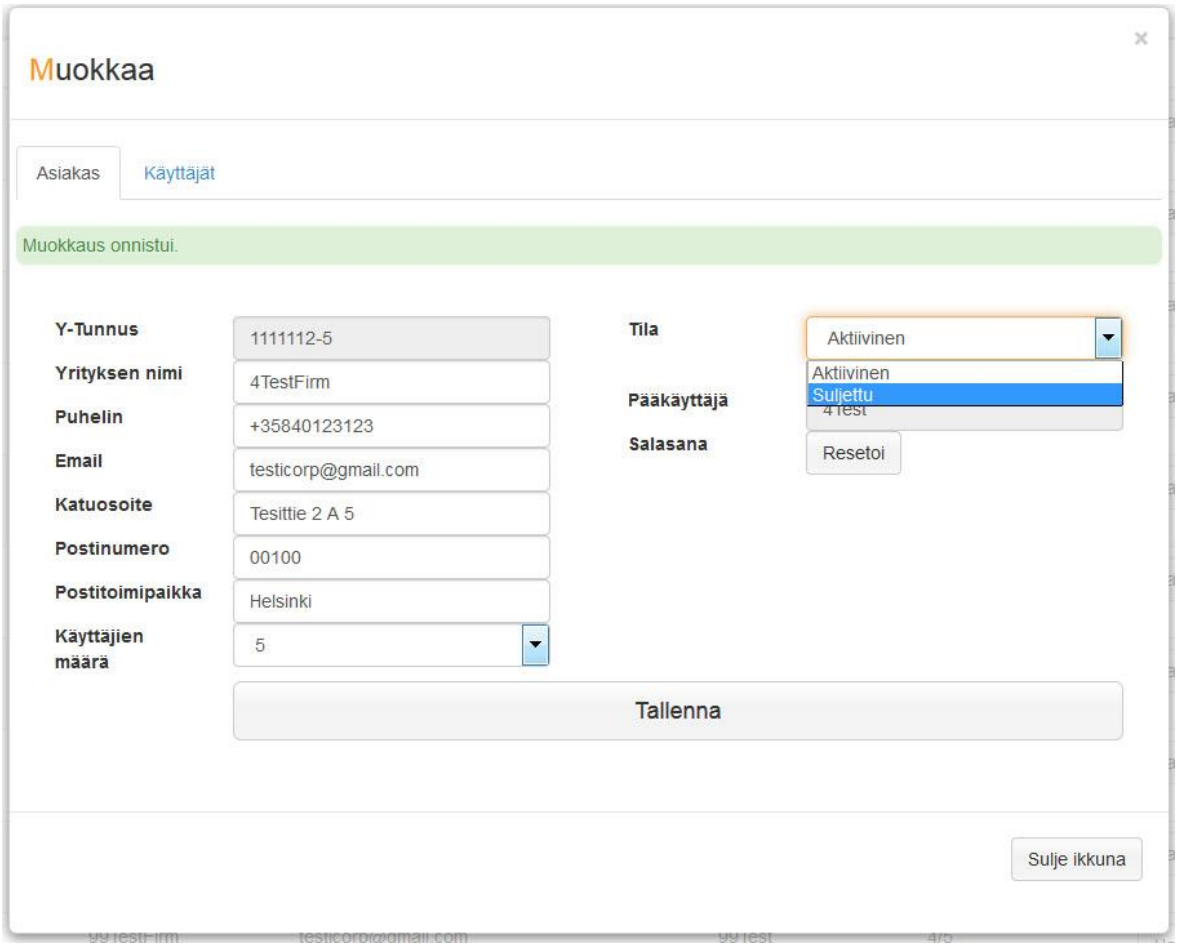
Asiakkaita päästään lisäämään, kun klikataan ”lisää”-nappia (kuva 3), jolloin näkyville tulee uusi ikkuna (kuva 4). Ikkunassa on vaaditut kentät, jotka validoidaan vaaditusti ensin jQuery:n avulla ja sitten hyödyntäen bean-validointia. Virheellinen syöte antaa välittömästi käyttäjälle virheilmoituksen, joka vastaa virheellisen syötteen syytä. ”Tallenna”-nappi lisää asiakkaan sekä käyttäjän tietokantaan ja antaa käyttäjälle vaaditut oikeudet.

Kuva 4. Asiakkaan lisääminen



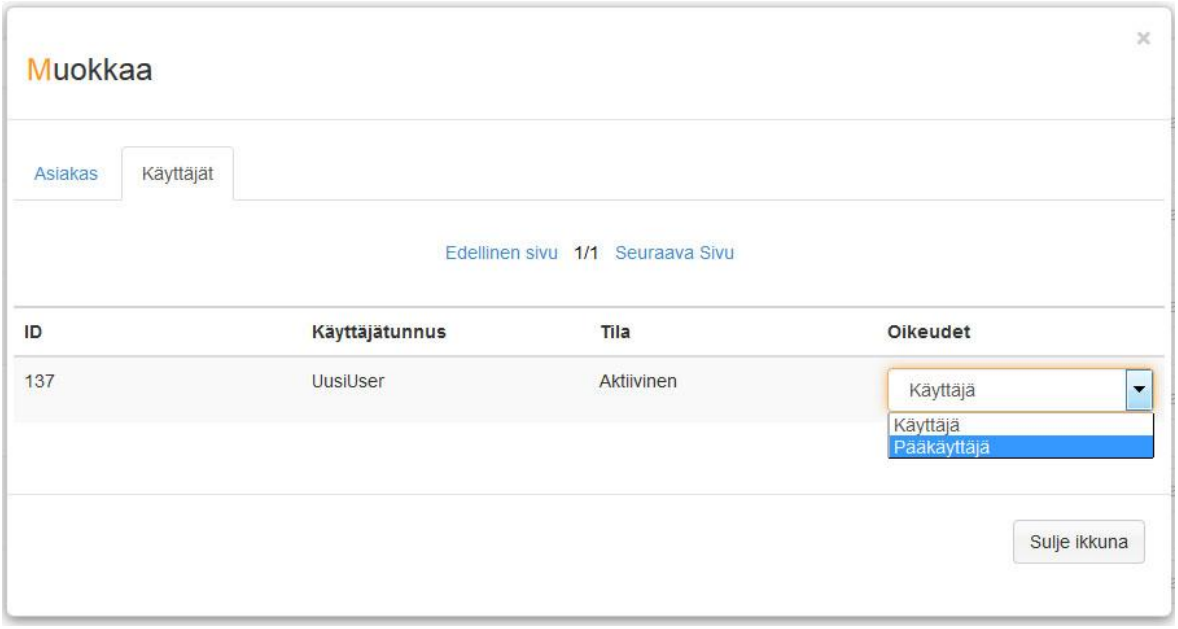
Kuvassa esitellään (kuva 5) asiakkaan hallinnointia, joka tapahtuu klikkaamalla ”hallinnoi”-nappia (kuva 3) halutun asiakkaan kohdalla. Hallinnassa asiakkaan tietoja voidaan muokata, käyttäjien määrää lisätä, tili voidaan sulkea ja pääkäyttäjän salasana voidaan resetoida. Virheellinen syöte antaa myös muokkauksen yhteydessä käyttäjälle ilmoituksen. Onnistunut muokkaus kertoo käyttäjälle muokkauksen onnistumisesta virheällä pohjalla varustetulla viestillä.

Kuva 5. Asiakkaan muokkaaminen



Kuvassa (kuva 6) listataan asiakkaan käyttäjiä. Listassa näkyy vain lisätyt käyttäjät ja pääkäyttäjä on jätetty ulkopuolelle, koska pääkäyttäjän oikeuksia ei voi muokata. Käyttäjän oikeuksia voi muokata vaihtamalla kenttään toinen arvo. Kentässä on kaksi oletusarvoa, jotta on yksinkertaista valita mitkä oikeudet haluaa määritellä kullekin käyttäjälle.

Kuva 6. Käyttäjien listaaminen ja oikeuksien muuttaminen



Pilottisovellus kykenee nyt turvallisiin maksusuorituksiin projektin toimeksiantajan verkkosivun kautta hyödyntäen ajan tasalla olevaa HMAC-menetelmää ja tukee siten yrityksen liiketoimintaa. Jokainen tehty metodi hyödyntää virheen ilmaantuessa tehtyä luokkaa, jonka kautta lähetetään automaattisesti luotu virheraportti projektin toimeksiantajan sähköpostiosoitteeseen.

3.9 Yhteenveto

Olemassa olevan sovelluksen jatkokehitys on merkittävää sinällään, että se on viimeinen suuri askel yritykselle ennen sen markkinoille vientiä. Etäältä tehty työ rajoittuu projektin toimeksiantajan toiveiden mukaan siten, että projektissa toteutettiin asiakkaiden ja käyttäjien hallintaan liittyviä ominaisuuksia, virheenkäsittelyä ja verkkosivujen sekä maksupalvelun yhdistämistä osaksi sovellusta. Jatkokehitys toteutettiin Javalla sekä JavaScriptillä hyödyntäen jQuery-, JSON- ja AJAX-tekniikkaa.

Ominaisuuksien määrittäminen nähtiin myös todella tärkeänä, koska se nähtiin edellytyksenä projektin onnistumiselle. Tarkasti määritellyt ominaisuudet mahdollistivat ketterän sovelluskehityksen ja pienensivät virhemarginaalia. Testaus olikin varsin onnistunut tästä syystä, vaikka se jäikin varsin vähäiseksi.

Projekti oli todella laaja ja siihen kului paljon aikaa ja maksujärjestelmä nähtiin hankalimpana osa-alueena, koska esimerkkejä ei ollut saatavissa ja laaja pohjatieto tarvittiin. Mak-

sujärjestelmästä saatiin tavoitteiden mukaisesti turvallinen. Opinnäytetyö oli haaste ketterälle sovelluskehitykselle ja sen onnistuminen tulee todennäköisesti auttamaan minua myös työelämässä.

4 Pohdinta

Projektin tuloksena syntyi kompakti käyttöliittymä asiakkaiden hallinnoinnille, oikeuksien hallinnoinnille ja turvallisten maksujen suorittamiselle. Toivotut ominaisuudet on toteutettu onnistuneesti ja ilmaantuvat virheet lähetetään automaattisesti eteenpäin sähköpostilla.

4.1 Tulosten tarkastelu

On tärkeää saada näkökulma myös toimeksiantajan osalta, jotta voidaan punnita onnistumista. Projektin toimeksiantajalle lähetettiin liuta kysymyksiä, joiden vastauksia on hyvä pohtia. Projektin toimeksiantajalle esitettiin seuraavat kysymykset:

- Miten projekti sujui?
- Toimiko kommunikointi?
- Tuliko toivotut ominaisuudet toteutettua?
- Onko lopputulos turvallinen?
- Toteutuiko toimeksianto aikataulussa?
- Onko lopputulos toimiva jatkokehityksen kannalta?
- Olisiko parantamisen varaa? Mitä?

Haastattelun tulokset löytyvät liitteestä (liite 5).

4.2 Testauksen merkittävyys

Projektin laajuuden vuoksi testausta ei harjoitettu kovin laajasti vaan se jäi varsin vähäiseksi. Testausta suoritti projektipäällikkö ja projektin toimeksiantajan testaaja, joten testitiimi on varsin pieni. Testaustulokset (liite 2) osoittavat, että kaikki vaaditut ominaisuudet toimivat niin kuin niiden pitääkin. Laajemmassa testauksessa poikkeuksia olisi saattanut ilmetä enemmän.

4.3 Johtopäätökset

Projekti edistää merkittävästi pilottisovelluksen vientiä markkinoille, koska pilottisovellus kykenee nyt turvallisiin maksusuorituksiin ja tukee siten yrityksen liiketoimintaa. Kompakti asiakkaiden hallinta, automaattinen virheiden raportointi ja turvallinen maksusuoritus auttavat pilottisovelluksen jatkokehitystä ja sen lopullista viemistä markkinoille.

4.4 Jatkokehittäminen

Asiakkaiden listaaminen on melko työlästä, kun asiakkaita on paljon ja niitä joutuu selaamaan usean sivun verran läpi, jotta löytää etsimänsä. Asiakkaiden hakua voisi helpottaa siten, että niitä haettaisiin valitulla hakukriteerillä tai Y-tunnuksella. Hakukentän voisi sijoittaa samaan laatikkoon, josta asiakkaita lisätään, jolloin se ei olisi myöskään tiellä.

Käyttäjän oikeuksia voi tällä hetkellä hallinnoida, mutta käyttäjiin ei voi muulla tavalla vaikuttaa. Ongelmatilanteessa muun kuin pääkäyttäjän tilin resetointi saattaa tulla tarpeeseen. Ylipäätänsä käyttäjän tietojen muokkaaminen saattaisi olla käytännöllistä siltä kannalta, että asiakkaan puolella ilmenee ongelmia.

Asiakkaan voi nyt lisätä rekisteriin manuaalisesti, mutta se ei lähetä asiakkaalle laskua tai luo tilausta tietokantaan. Tällaiset tiedot olisi varmasti hyvä olla jos asiakkaan joskus joutuu lisäämään manuaalisesti. SERIALIZABLE isolation-tasona on varmasti ehdoton näin tärkeissä tiedoissa, jotta tilaus ei katoa mihinkään.

Maksuprosessin olisi hyödyllistä palauttaa asiakkaalle virhekoodi nähtäväksi, jotta tämä voi raportoida ongelman. Virhekoodi olisi myös käytännöllistä lähettää eteenpäin, jotta myös projektin toimeksiantajalle jää siitä jonkinlainen merkintä. Tilauksen luonti tietokantaan maksuprosessin yhteydessä hyödyttäisi myös tilausten seurantaa. SERIALIZABLE isolation-tasona on varmasti ehdoton näin tärkeissä tiedoissa, jotta tilaus ei katoa mihinkään.

Asiakkaan olisi hyvä päästä katsomaan tilaustietojaan verkkosivulta, kun asiakastili on luotu maksun yhteydessä. Tiedoissa voisi näkyä esimerkiksi tilauspäivä, onko tilaus voimassa, käyttäjien määrä ja laskutusosoite. Lisäksi laskuja on helppo hallinnoida tällaisen palvelun kautta, jossa voidaan esimerkiksi listata laskut ja luoda laskun viereen nappi, josta pääsee maksamaan laskun.

4.5 Ammatillinen kehittyminen

Projekti oli todella laaja ja siihen kului paljon aikaa. Hankalin osa-alue oli HMAC-arvon luonti, koska Java-pohjaisia esimerkkejä maksuprosessille ei ollut olemassa ja koko maksuprosessin ymmärtäminen vaati laajaa pohjustusta. Laaja pohjatieto maksuprosessista auttoi tietoturvan tavoitetason pitämisessä. Testaamaan ei myöskään kovin usein päässyt opintojen ohella eikä menetelmiä testaamiseen juurikaan kerrottu, joten testaaminen oli mielestäni melko vaativaa.

Näin opinnäytetyön alun perin jo jonkinlaisena haasteena ketterälle sovelluskehitykselle. En asettanut haasteita aivan mahdottomalle tasolle, mutta sain silti projektin toteutettua kaksi viikkoa etuajassa ja koen ylittäneeni itseni. Luulen asettamani haasteen auttavan minua työelämässä, koska nyt minulla on, jotain mistä kertoa, kun minulta kysytään ketterästä sovelluskehityksestä.

Liikkeelle lähdettiin siitä, että projekti tulee saada ajoissa valmiiksi, jotta ei tule taloudellisia tappioita. Tehtävien priorisointi ei tuottanut ongelmia, koska tarvittavat ominaisuudet oli helposti pääteltävissä. Projektia priorisoitiin käytännössä sen mukaan, että sovellus saadaan mahdollisimman pian markkinoille.

Lähteet

Peltomäki, J. 2000. Javascriptin peruskirja. 2. painos. Teknolit. Jyväskylä.

Otero, C., Larsen, R. Professional jQuery. Wiley. Canada

Asleson, R.,Schutta, .N, T. 2007. AJAX - tehokas hallinta. Readme.fi, Jyväskylä

Bell, D., Parr, M. 2005. Java for students. 4. painos. Pearson.

Gates, B. Writing secure code. 2. painos. Microsoft press.

Patton, R. 2005. Software Testing. 2. painos. Sams publishing.

Kofler, M. The Definitive Guide to MySQL. 2. painos. Apress

Information Technology Laboratory 2014a. Secure Has Standard (SHS). Luettavissa: http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf. Luettu 13.11.2014

Information Technology Laboratory 2014b. The Keyed-Hash Message Authentication Code (HMAC). Luettavissa: http://csrc.nist.gov/publications/fips/fips198-1/FIPS-198-1_final.pdf. Luettu 13.11.2014

Information Technology Laboratory 2014c. Luettavissa: http://csrc.nist.gov/publications/nistpubs/800-107-rev1/sp800-107-rev1.pdf. Luettu 13.11.2014

JSON 2014a. Introducing JSON. Luettavissa: http://www.json.org/. Luettu 05.11.2014

JSON 2014b. JSON in JavaScript. Luettavissa: http://www.json.org/js.html. Luettu 05.11.2014

Oracle 2014a. What Is an Object? Luettavissa: http://docs.oracle.com/javase/tutorial/java/concepts/object.html. Luettu 05.11.2014

Oracle 2014b. Defining Methods. Luettavissa:
https://docs.oracle.com/javase/tutorial/java/javaOO/methods.html. Luettu 11.11.2014

Oracle 2014c. Lesson: Annotations: Luettavissa:
https://docs.oracle.com/javase/tutorial/java/annotations/. Luettu 17.11.2014

Spring 2014a. Web MVC framework. Luettavissa:
http://docs.spring.io/spring/docs/current/spring-framework-reference/html/mvc.html. Luettu 28.10.2014

jQuery. .serialize(). Luettavissa:
http://api.jquery.com/serialize/. Luettu 13.11.2014

Validate. Validate forms like you've never validated before!. Luettavissa: http://jqueryvalidation.org/documentation. Luettu: 13.11.2014

MySQL 2014a. Information Functions. Luettavissa:
http://dev.mysql.com/doc/refman/5.0/en/information-functions.html#function_row-count. Luettu: 13.11.2014
MySQL 2014b. Data Manipulation Statements. Luettavissa:
http://dev.mysql.com/doc/refman/5.0/en/sql-syntax-data-manipulation.html. Luettu 13.11.2014

Spring 2014b. Spring Framework. Luettavissa:
http://projects.spring.io/spring-framework/. Luettu 28.10.2014

Spring 2014c. Introductions to Spring Framework. Luettavissa:
http://docs.spring.io/spring-framework/docs/current/spring-framework-reference/html/overview.html#overview-modules. Luettu 28.10.2014

Spring 2014d.Transaction Management. Luettavissa:
http://docs.spring.io/spring-framework/docs/current/spring-framework-reference/html/transaction.html. Luettu 12.11.2014

Spring 2014e. Validation, Data binding, and Type Conversion. Luettavissa:
http://docs.spring.io/spring/docs/current/spring-framework-reference/html/validation.html. Luettu 13.11.2014

W3Schools 2014a. HTML Forms and Input. Luettavissa: http://www.w3schools.com/html/html_forms.asp. Luettu 28.10.2014

PuTTY. PuTTY FAQ. Luettavissa: http://www.chiark.greenend.org.uk/~sgtatham/putty/faq.html#faq-what. Luettu 06.11.2014

Liitteet

Liite 1. AJAX-kutsu

(Salainen)

Liite 2. Testaustulokset

(Salainen)

Liite 3. Aikataulu

(Salainen)

Liite 4. Product Backlog

(Salainen)

Liite 5. Haastattelun tulokset

(Salainen)